Diversification Catalyzes Language Models' Instruction Generalization To Unseen Semantics

Dylan Zhang
University of Illinois Urbana-Champaign
shizhuo2@illinois.edu

Justin Wang
University of Chicago
jw93@illinois.edu

Francois Charton
Meta
fcharton@meta.com

Abstract

Instruction-tuned language models excel in knowledge, reasoning, and instruction-following. While knowledge and reasoning are well-explored, the factors enabling generalization to unseen instructions remains underexplored due to challenges in isolating instruction-following dynamics.

In this work, we model instruction-following as a computational process and design controlled experiments inspired by the Turing-complete Markov algorithm to disentangle its dynamics. Our findings reveal that the ability to generalize to instructions with unseen semantics emerges only when training data is strategically diversified across rich semantics. This finding gives us the hammer that breaks down the wall separating training instructions from unseen ones encountered in the wild. For specialist models, a balanced mix of in-domain and diverse out-of-domain tasks enhances performance more effectively than simply increasing in-domain data. For generalist models, domain diversification consistently outweighs the costs of reduced task-specific data, regardless of data budgets. Furthermore, we show that proper diversification with a lower data budget can outperform simply scaling up data volume. These findings highlight strategic data diversification as key to optimizing instruction-following and improving model performance across applications.

1 Introduction

The rapid advancements in large language models (LLMs) have revolutionized a wide range of tasks, including language comprehension (Wang et al., 2020), generation (Brown et al., 2020), knowledge-based question answering (Hendrycks et al., 2021a), and solving complex reasoning problems in fields such as mathematics (Cobbe et al., 2021; Hendrycks et al., 2021b) and programming (Chen et al., 2021a; Austin et al., 2021; Chen et al., 2021b; Li et al., 2022). Instruction-tuning plays an important role in making language models performing these tasks.

The current research landscape regarding instruction tuning data has yielded varied and sometimes contradictory findings on composition (Zhou et al., 2023a) and scaling (Zeng et al., 2024; Zhang et al., 2024a) of instruction tuning data. These inconsistencies suggest that the best practices to craft an instruction tuning dataset is not fully understood (Dong et al., 2024; Zhang et al., 2024b). A key limitation is the absence of controlled studies that systematically examine different aspects of LLM capabilities, where the effects of instruction tuning data may follow different patterns. While reasoning (multi-step deduction, like math problem-solving) and knowledge retrieval (accessing information stored in the parameters) have been extensively studied, instruction following—essential for practical interaction with users (Zhou et al., 2023b; Liu et al., 2024a)—remains poorly understood. The role of instruction-tuning (Ouyang et al., 2022; Taori et al., 2023; Wei et al., 2022; Sanh et al., 2022a) data composition and diversity in enabling LLMs to generalize to unseen tasks is particularly unclear.

Our work addresses this gap by focusing explicitly on instruction following capabilities of LLMs. To address this gap, our work presents a systematic investigation into instruction-following capabilities, leveraging controlled experiments inspired by the Turing-complete Markov algorithm (Markov, 1954).By isolating instruction-following, we reveal that diverse cross-domain instruction data is decisive for enabling LLMs to generalize effectively to unseen semantic tasks. Our findings reveal that robust generalization emerges only when instructions span diverse, cross-domain semantics. Limited domain diversification, in contrast, fails to achieve this goal, underscoring the necessity of training strategies that prioritize broader instruction diversity. Our findings reveal that diversification confined to limited domains does not guarantee robust generalization. In contrast, cross-domain diversification significantly enhances the model's adaptability to new instructions, highlight-

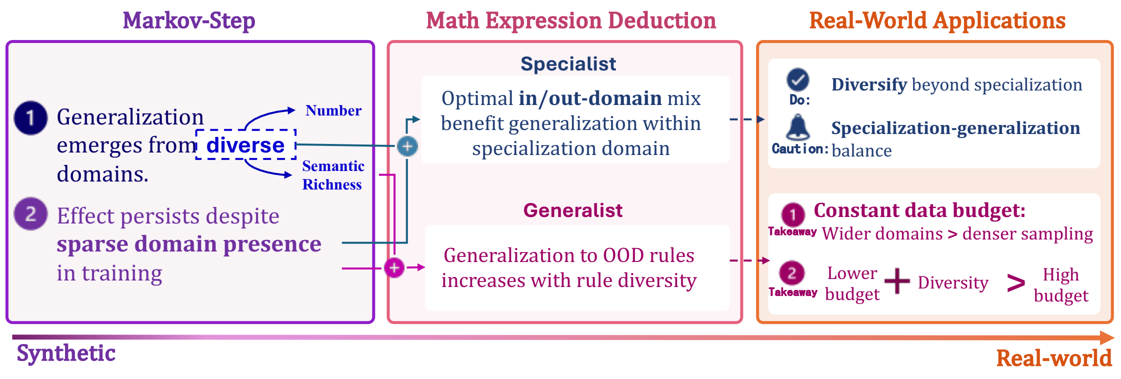

Markov-Step
Math Expression Deduction
Real-World Applications
1 Generalization emerges from diverse domains.
Number
Semantic Richness
2 Effect persists despite sparse domain presence in training
Specialist
Optimal in/out-domain mix benefit generalization within specialization domain
Generalist
Generalization to OOD rules increases with rule diversity
Do: Diversify beyond specialization
Caution: Specialization-generalization balance
1 Constant data budget:
Takeaway Wider domains > denser sampling
2 Takeaway Lower budget + Diversity > High budget
Synthetic
Real-world

Figure 1: A mindmap of our study and key takeaways.

ing the importance of a more diverse training strategy.

Our research offers actionable insights into dataset collation for both **specialist** and **generalist** LLMs. For specialist models (e.g., code language models), expanding data diversity beyond core domains enhances instruction-following performance, but requires balancing specialization and diversification. For generalist models, cross-domain data mixtures improve generalization to unseen instruction semantics. We identify two key principles: (1) increasing dataset diversity, even without adding volume, outperforms reliance on established datasets, and (2) diversifying instruction semantics is more effective than scaling within the same distribution.

Our study establishes that instruction-following capabilities in LLMs hinge on strategic cross-domain diversification, suggesting best practices for crafting instruction-tuning datasets to unlock robust and adaptable generalization.

2 Abstracting Instruction-Following As String-Replacement

2.1 Markov-Step: Conditional Sub-string Replacement Task

We aim to first simplify instruction-following to its core components, allowing us to systematically study the role of data diversity in isolation.

We model such tasks as string-replacement operations, a foundational concept in theoretical computer science. Our string replacement set-up underpins Markov Algorithms (Markov, 1954), a Turing-complete framework where sequences are iteratively transformed using ordered rewrite rules. These algorithms follow a structured, rule-driven process: the first applicable rule replaces the leftmost matching sub-string, and the process continues until no further matches are found. This stepwise transformation reflects the sequential, instruction-driven behavior that we aim to study. Appendix A includes illustrative examples demonstrating Markov algorithms.

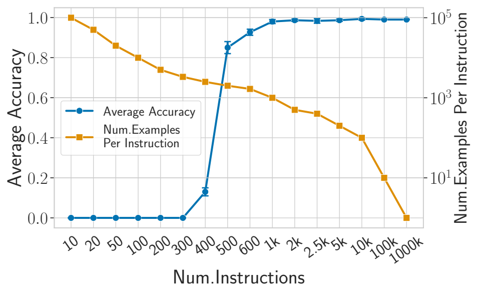
In our study, we focus on a simplified form of this process, which we call Markov-Step. The replacement rule R—a pair like $aa \to bac$—is applied to an input string ξ (e.g., $caaba$), yielding an output string τ (e.g., $cbacba$). The rule is applied to the leftmost occurrence of a match, and if no match exists, the original input remains unchanged.

This task, though simple, serves as a powerful proxy for instruction tuning by teaching models to handle structured, rule-based transformations. A generalization of this task will be introduced in Section 3, where x and y represent abstract patterns (e.g., $x = a^2$ matching all squared terms). This task is a step towards our ultimate goal of studying effects of data on pre-trained models' ability to follow instructions.

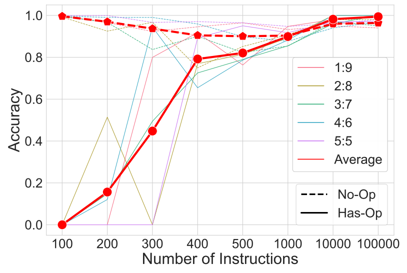
2.2 Experiment Set-Up

We start from a very simple scenario where the model applies a rule $\iota : x \to y$ to an input string ξ, replacing the leftmost occurrence of x with y, if $x \subset \xi$ (x is present in ξ). If x is not present, the input remains unchanged. This subsumes the essence of instruction-following for models: learn from input-instruction-output tuples, given a pair of input and instruction, **whether** and **how** to correctly perform the desired transformation.

We train decoder-only transformer models from scratch for the controlled experiments in this section, avoiding biases from pre-training. The training dataset consists of $S \times I$ sequences, where S is the number of input strings and I the number of replacement rules. Models are tested on 10^5 unseen examples to assess generalization. Full training details are in Appendix B.



(a) Re-writing accuracy against the number of instructions with a fixed-size training set.



(b) Rewriting with no-op situation included.

Figure 2: Generalization versus the number of instructions during training.

2.3 Results

Instruction Diversity Is Decisive To Generalization Figure 2a presents the generalization accuracy for models trained on a fixed budget of $I \times S = 10^6$ examples with varying I and S where all sequences contains pattern to find. We observe a sharp step-

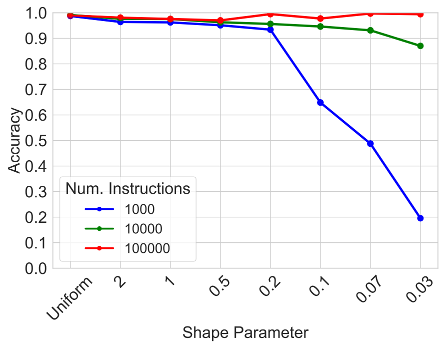


Figure 3: Effect of long-tail task distributions on model's generalization ability.

shaped transition: models trained on fewer than I_{min} (where $I_{min} = 300$) unique instructions consistently fail to generalize, regardless of example count per instruction. Conversely, models exposed to over I_{max} (where $I_{max} = 1{,}000$) distinct instructions generalize effectively to unseen instructions, even with very few examples per instruction, since the budget $S \times I$ is fixed. This clear phase-transition proves that within a constant data budget, large enough number of distinct instructions (I) is a key driver for generalization underpinning the necessity of cross-domain diversification for robust instruction-following for unseen instructions (which will be shown in later sections).

A slightly more complex situation involves dealing with no-ops. Figure 2b presents the generalization accuracies of trained models, as a function of the number of instructions and the frequency of No-Ops. Interestingly, despite No-Ops dominating the dataset[1], the model generalizes well to unseen instructions after training on around 400 distinct cases. The proportion of No-Ops does not significantly affect generalization beyond that point - the model learns whether a transformation is needed at all only if the instruction semantics is diversified beyond the threshold.

Imbalanced Distribution Is Still Effective In Driving Generalization In earlier experiments, training sets evenly distributed instructions across examples: a set of 1 million examples with 500 instructions featured each instruction 2000 times. However, real-world datasets are rarely uniform; certain tasks dominate due to data availability and task nature.

To study how instruction distribution affects generalization to unseen tasks, we created datasets with 1,000, 10,000, and 100,000 unique instructions. Examples per instruction followed a power law distribution defined by $f(x) = \alpha x^{\alpha-1}$ where α is the shape parameter. By varying the shape parameter of the power law, we can generate a distribution of examples that range from close to uniform, to extremely peaked as shown in Fig. 8.

Figure 3 shows model generalization as a function of the power law's shape parameter for training sets of $N = 1$ million examples with $I_1 = 1{,}000$, $I_2 = 10{,}000$, and $I_3 = 100{,}000$ instructions. Models trained on I_2 or more instructions are robust to the distribution of examples per instruction. For models trained on I_1, generalization accuracy drops sharply when the shape parameter exceeds $\alpha = 0.2$. In such cases, instructions with a probability lower than $p_{low} = 0.1\%$ are barely represented, and the model effectively trains on fewer than $I_{min} = 500$ instructions, the minimum for generalizing to unseen instructions.

These results suggest that sufficient semantic coverage—not uniform distribution—is key for generalization. Broad coverage across tasks, even with imbalanced data, can enable effective model generalization.

Semantic Diversification Boosts Task Performance In practical instruction-tuning, uniform sampling

[1]Consider a dataset containing 100,000 data points, 10% No-Ops, and 100 rules. No-Ops takes up 10,000 in total, $\sim 11\times$ of any has-Ops.

across all possible instruction semantics is infeasible. To emulate real-world constraints, we trained models on semantically restricted patterns: **repeated characters** (*aaabbbccc* for $k = 3$ - each character repeated 3 times), **periodic patterns** (*abcabc* for $k = 2$ a sub-string repeated 2 times), and **mirrored structures** (*abccbaabc* for $k = 3$ mirroring the sub-string for 3 times), and measure their generalization across different levels of k, where k is a parameter controlling the constrained-ness of rule semantics. Our findings show that models trained on a single

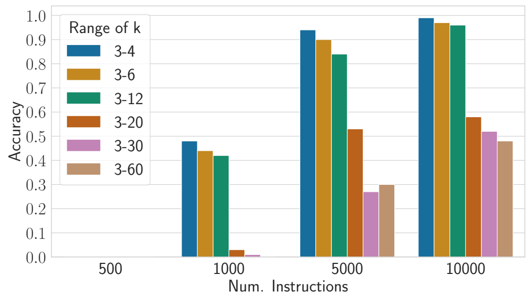


Figure 4: Model's performance on $k < 3$ when trained on the three classes of restricted semantics as in 2.3. Models trained on 500 or less instructions never generalize to smaller k.

constrained sub-domain with high constraints (large k) fail to generalize to less constrained tasks (low k). Training on mixtures of constrained sets improved generalization only when the instruction space was sufficiently diverse. Larger instruction sets (e.g., 5000 examples) boost generalization, while highly restricted semantic domains (larger k) make it harder.

Our findings highlight the importance of spanning semantically diverse subspaces to foster robust generalization, rather than relying on large datasets within narrowly defined domains.

3 Rewriting with Abstraction

Real-world instruction-following often requires abstracting high-level concepts and grounding them in specific contexts. To simulate this, we extend the string-rewriting experiment by introducing abstraction: **abstract rules** describes the general patterns

We design a mathematical deduction task as a context-sensitive generalization of the Markov string replacement task (Section 2). The task involves simplifying algebraic expressions using specified deduction rules, mimicking instruction-following across semantic domains. We ensure it remains challenging for pre-trained models (see **ZS** and **FS** performances in Figure 5a) to eliminate confounding factors.

In this experiment, we present the model with a randomly generated mathematical expression I and an **abstract** mathematical deduction rule $\iota_{abs} : (X = Y)$ where X and Y are math expressions (e.g. an example abstract rule would be $a^2 - b^2 = (a + b) \times (a - b)$, to assess the model's ability to identify the relevant sub-expression in ξ (e.g. in $(2x+5)^2 - (3y-6)^2$, here $a = 2x+5, b = 3y-6$) and correctly apply the transformation (get $(2x+5+3y-6) \times (2x+5-(3y-6))$). We observe how well the model generalizes when trained on datasets of varying rule diversity.

A full example will be:

`Rule:` $a^2 - b^2 =$ (a + b) $\times$ (a - b)

`Input:` $((2x+5)^2 - (3y-6)^2)^3 + (\log(5t) - \cos 4k)$

`Output:` ((2x + 5 + 3y - 6) $\times$ (2x + 5-(3y - 6)) $)^3 + (\log(5t) - \cos 4k)$

3.1 Data Generation

We collate a set of distinct **equational** algebraic deduction rules of the form LHS = RHS.

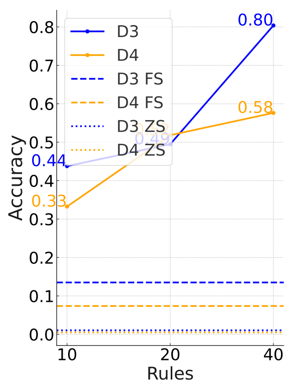
Random Tree Generation We randomly construct mathematical expression trees of a specified depth d. Non-leaf nodes were systematically assigned operators (e.g., $+$, $-$, $*$, $/$), while leaf nodes were populated with variables, constants, or unary operations.

Producing Pattern-Carrying Sub-Expressions To generate expressions, a pattern-carrying sub-tree was generated with a depth of d_p, denoting the depth of expression tree for each entry in the rule. e.g. we may replace a with $(y+2x+5)$ with $d_p = 2$. We then randomly choose a leaf node and swap it with the **concrete** sub-expression.

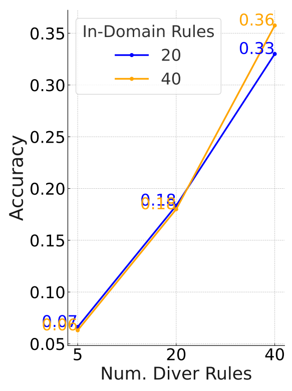
3.2 Simulating Specialist and Generalist

We examine two common settings for instruction-tuning: ***generalist***, and ***specialist*** training via the set of simulated experiments. We fine-tune all models from a pre-trained Mistral-7B-v0.3 (Jiang et al., 2023) checkpoint.

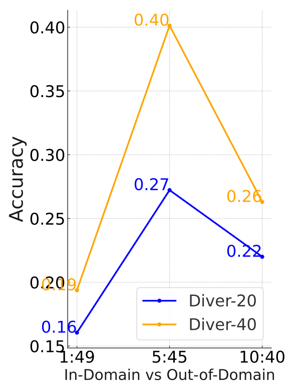
Diversifying Instruction Semantics Empowers Better Generalist LMs We control the number of training instances (50K), and vary the number of abstract deduction rules & the instantiated rules per abstract rule. We test the models on unseen sequences and unseen abstract rules. We train the model on triples of (ξ, ι_{abs}, τ) pairs where τ is the result of applying ι_{abs} to ξ. The result is shown in Figure 5a.Our findings demonstrate a clear advantage of increased rule diversity, consistent with previous string rewriting experiments. Holding the number of training instances constant, we observed that expanding the diversity of



(a) Accuracy On Unseen Deduction Rules vs. Abstract Rule Diversity. **D3**:pattern with tree depth 3. **ZS**:Zero shot. **FS**: Few shot.



(b) Accuracy on unseen depths against number of diversification rules with the same in-domain / out-of-domain mixture.



(c) Accuracy on unseen depths vs. In-domain/out-of-domain combination. **Diver-** means number of diversification rules. X axis ticks are $|\mathcal{R}_{diver}|$: $|\mathcal{R}_{spec}|$.

Figure 5: Rewriting With Abstraction.

rules (ι) in the training data significantly enhances the model's ability to generalize to unseen abstract rules during testing. This improvement is achieved despite the model encountering fewer groundings per rule and expression. This extends our results from Markov-step experiments to pre-trained language models with rules that require abstraction and grounding of commands, a closer resemblance of actual use-case scenarios.

Sweet Point Between Specialization and Diversification For Optimal Specialist Performance In this experiment, we simulate a scenario where a model is trained as a specialist and tested on out-of-distribution queries that still belong to the same overall instruction type. To achieve this, we divide the set of rules $\mathcal{R}$ into two categories: specialized rules $\mathcal{R}_{\text{spec}}$ and diversification rules $\mathcal{R}_{\text{diver}} = \mathcal{R} \setminus \mathcal{R}_{\text{spec}}$. The training data is constructed as a mixture of instances generated from these two sets of rules. Specifically, the instances based on the specialized rules use patterns with depth $d_p^{\text{train}} < d_p^{\text{test}}$ to simulate out-of-distribution test instances, while the instances from the diversification rules add variety to the training. This setup allows us to examine the trade-off between specialization and diversification for better instruction following in this specialized task.

As shwon in Figure 5c, the results exhibit a clearly peaked structure as we incorporate more out-of-domain data for diversification. This reflects a sweet-point between specialization and enhanced instruction-following via training on a more diverse set of instructions. Figure 5b shows the trend when we diversify across an increasingly rich semantics. We notice the benefit of a more diverse $\mathcal{R}_{diver}$, which suggests that even when diversifying for specialists, one should be mindful to curate a dataset that spans over wider domains.

4 Specialist Instruction Follower - Case Study Of Code Generation

In this section, we investigate the specialist training scenario, where the objective is to train a language model to adapt to a certain task domain. Specifically, we consider code generation task as an example since it features instruction following aspect - requiring conversion of natural language prompts into executable code, demanding precise interpretation and faithful execution of input instructions over complex reasoning or knowledge.

4.1 Experiments

We evaluate on two widely-used code generation benchmarks: HumanEval (Chen et al., 2021a) and MBPP (Austin et al., 2021), alongside the augmented EvalPlus (Liu et al., 2023) for evaluation. These benchmarks present a diverse collection of coding problems that test the model's ability to interpret and execute novel instructions. The training instances come from OSS-Instruct (Wei et al., 2023), a synthetic coding dataset, which has been sanitized to avoid data contamination.

We utilize two state-of-the-art pre-trained code language models as base models: DeepSeek-Coder-6.7B-Base (Guo et al., 2024) and CodeQwen-1.5-7B-Base (Bai et al., 2023).

4.2 Proper Diversification Yields Better Coders

The Role of Semantic Diversity in Generalization Our results in Tables 2 and 1 reveal a critical insight:

Base Budget (OSS)	+OSS	+Alpaca	+CoT	HE	HE+	MBPP	MBPP+	Avg (Base)	Avg (+)	Avg	Rel. Gain
15K	5K	0	0	62.2	56.7	75.7	62.4	68.9	59.5	64.2	-
15	4	1	0	67.1	59.8	75.7	62.2	71.4	61	66.2	3.1
15	3	2	0	**68.3**	**61.6**	75.4	63.2	**71.9**	**62.4**	**67.1**	**4.6**
15	2	3	0	64.6	60.4	76.4	63.4	70.5	61.9	66.2	3.1
15	1	4	0	65.2	58.5	**76.7**	**63.7**	71	61.1	66	2.8
15	0	5	0	64.6	57.9	73.9	61.4	69.3	59.7	64.5	0.5
60	15	0	0	66.5	61.6	75.4	61.9	71	61.8	66.4	-
60	0	15	0	67.1	59.8	77	64.8	72.1	62.3	67.2	1.2
60	0	7.5	7.5	64.6	59.1	76.2	63.8	70.4	61.5	65.9	-0.6
60	7.5	7.5	0	68.9	61.6	76.2	63.8	72.6	62.7	67.6	1.9
60	7.5	3.25	3.25	66.5	**62.2**	76.2	64.3	71.4	63.3	67.3	1.4
60	12.5	2.5	0	68.3	61	76.2	64.3	72.3	62.7	67.5	1.6
60	12.5	1.25	1.25	68.3	**62.2**	**77.8**	**65.1**	**73.1**	**63.7**	**68.4**	**3**
60	14	1	0	**69.5**	**62.2**	76.2	64.3	72.9	63.3	68.1	2.5
60	14	0.5	0.5	66.5	61	76.5	63.2	71.5	62.1	66.8	0.7

Table 1: Results on DeepSeek-Coder-6.7B and comparison With MagiCoder-DS-6.7B (Wei et al., 2023). Plum-colored row surpasses the performance of full-data training. Best configurations corresponding to each setting are highlighted. We demonstrated that one could achieve higher performances by means of diversification.

increasing the size of coding datasets alone is not always the most effective strategy for improving performance on coding tasks. Instead, **diversifying instruction domains yields superior results**. For instance, incorporating data from general-purpose QA (Alpaca) and reasoning tasks (CoT) delivers significantly better outcomes than adding an equivalent volume of coding data. Crucially, the **Plum-colored** configuration in Table 1, which uses just 20,000 diversified data instances, achieves performance levels that surpass models trained on 75,000 OSS-Instruct data. This demonstrates that a carefully curated, diverse dataset can drive substantial improvements over merely scaling dataset size within a single domain.

This pattern is consistent with our findings across multiple experiments. In Section 3.2, we demonstrated that diverse instructions improve generalization by enabling the model to handle out-of-distribution tasks derived from seen abstract instructions. Similarly, in Section 2.3, even basic string-replacement tasks highlighted the value of diverse instructions. Crucially, these improvements occurred despite the lower quantity of diverse instructions relative to the main task, as shown in the long-tail distribution scenario in Section 2.3. These results collectively underscore the importance of instruction diversity in achieving robust performance across both abstract and practical tasks, even in data-imbalanced scenarios.

The Power of Cross-Domain Diversification By incorporating datasets like Alpaca, which is designed for human language interaction, and the CoT-Collection (Kim et al., 2023), which challenges the model with complex reasoning tasks, we extend the model's exposure to diverse semantic spaces. This further improves the model's generalization

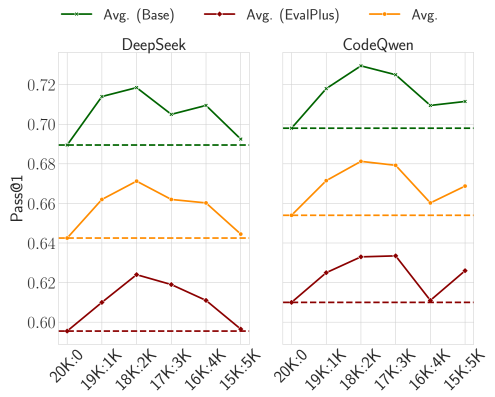


Figure 6: Sweet spots of Pass@1 with data mixture. Baseline is marked with dotted lines.

capabilities across domains. Results from CodeQwen, presented in Table 2 support this conclusion. Models trained on a balanced mixture of coding, general QA, and reasoning data outperform those trained solely on a mix of coding data and Alpaca, reinforcing the importance of cross-domain diversification for handling complex and varied instructions.

Balancing Generalization and Specialization While diversifying instruction data offers substantial benefits, there are limits. As shown in Figure 6, incorporating general-domain instructions initially boosts the model's ability to follow natural language specifications, leading to improved Pass@1 scores for code generation. However, as more non-coding data is added, the model's capacity to handle the nuanced requirements of coding tasks diminishes.

This trade-off echoes our findings from the synthetic experiments in Section 3.2: achieving optimal performance requires balancing general instruction-following capabilities with specialized domain knowledge (Ling et al., 2024). Figure 6 illustrates the plateau and eventual decline in performance, underscoring the importance of calibrating the mixture of coding and non-coding data to achieve the best overall results.

Guided by the findings of our earlier synthetic experiments, to improve **specialists**, we recommend strategic diversification beyond the current domain, while carefully balancing composition between in- and out-of task domain data.

5 Fine-tuning Generalist LLMs

In this section, we evaluate the impact of cross-domain instruction diversification on general reasoning tasks and investigate the optimal high-level strategy to improve the quality of a dataset.

5.1 Experimental Setup

In this study, we evaluate the impact of cross-domain instruction diversification on large language models (LLMs) by comparing our approach with a baseline model trained exclusively on UltraInteract-SFT dataset (Yuan et al., 2024). UltraInteract-SFT is a collection of complex, multi-step reasoning problems emphasizing on math problem-solving, code generation, and logical reasoning, promoting robust reasoning and planning capabilities in LLMs.

While UltraInteract-SFT primarily focuses on math and coding problems and contains a rich collection of those problems, its scope is limited to these domains. OpenOrca (Lian et al., 2023) and Alpaca, though sparse and varied, introduce broader instruction-following tasks.

We begin with varying initial data budgets from the UltraInteract-SFT dataset and systematically measure the performance improvements achieved by incrementally incorporating additional data points.

We gauged the model's overall capabilities using the same set of benchmarks consisting of coding (Austin et al., 2021; Chen et al., 2021a), math (Hendrycks et al., 2021b; Cobbe et al., 2021; Chen et al., 2023), knowledge (MMLU (Hendrycks et al., 2021a)), instruction following (Zhou et al., 2023b) and chain-of-thought reasoning (Suzgun et al., 2022) as Yuan et al. (2024) and computed average performance. To reflect on its precision in instruction following, we adopted IF-Eval (Zhou et al., 2023b) benchmark, comprising over 500 prompts for rigorous instruction-following tests. We followed a core-set selection approach when curating datasets of various budgets. We finetune from a pre-trained Mistral-7B-v0.3 checkpoint for all results in table 3. In the experiments, we study the effect of adding controlled quantities of different types of data to various pre-defined base budgets, and compare the performances across budgets to exhibit the advantage of dataset expansion along the dimension of improving diversity.

5.2 Data Diversity Matters More Than Quantity For Generalists

Table 3 provides compelling evidence for the critical role of diverse training data in enhancing model performance, as opposed to relying solely on volume.

Within-Budget Level. By simulating the incremental composition of training data from a base budget up to a predefined total budget, we systematically examine the impact of incorporating diverse datasets beyond **UI**'s domain. Our results show that the model consistently benefits more from this diversified composition than from simply expanding the **UI** dataset size.

Across Budget Levels. Performance comparison across different **Total** data budget levels reveals that models trained with diverse datasets even on lower budgets achieve superior generalization and task performance compared to those trained solely with additional data from the same distribution.

The results show that adding diverse datasets incrementally fosters generalization and robustness, even for reasoning tasks like math that typically rely on data volume (Yu et al., 2024). This reinforces the benefits of diversification highlighted in Section 3.2.

This analysis underscores a key insight: prioritizing diverse training datasets is crucial for building adaptable, high-performing models. Exposure to varied domains enhances generalization to unseen instructions, demonstrating that data diversity outweighs sheer volume in impact. Diversity is thus a foundational requirement for models to excel across complex and varied tasks.

6 Related works

Datasets for instruction-tuning. Many datasets for instruction-tuning have been proposed. The best quality is achieved for sets collated by human annotators (Khashabi et al., 2020; Ye et al., 2021; Sanh et al., 2022b; Wang et al., 2022; Longpre et al., 2023; Conover et al., 2023; Köpf et al., 2023), but their size is constrained by the cost of annotation. Alternative methods, which use large language models to generate instruction sets, have been proposed (Wang et al., 2023b; Honovich et al., 2022;

Base Budget (OSS)	+ OSS	+ Alpaca	+ CoT	HE	HE+	MBPP	MBPP+	Avg (Base)	Avg (+)	Avg	Rel.Gain Intra -Budget	Rel.. Gain wrt. No Diver.
19K	1K	0	0	65.9	59.8	73.7	62.2	69.8	61.0	65.4	-	-
19	0	1	0	67.7	61.6	75.9	63.4	71.8	62.5	67.2	2.8	2.8
18	0	2	0	69.5	63.4	76.4	63.2	73.0	63.3	68.1	-	4.1
18	0	1	1	68.9	63.4	77.4	64.2	73.2	63.8	68.5	0.6	**4.7**
16	0	4	0	65.2	58.5	76.7	63.7	71.0	61.1	66.0	-	0.9
16	0	2	2	68.3	61.6	77.2	63.7	72.8	62.7	67.7	2.6	3.5

Table 2: Pass@1 with CodeQwen-7B.

Base Data Budget	+UI	+OO	+AL	Total	Overall Avg.	IF Avg.	Overall wo IF	Overall. wo Math	Rel. Gain + 20K Data Budget	Rel. Gain Diver. w./In Budget
	10K	0	0		39.07	25.32	47.27	44.94	-	-
10K	0	10	0	**20K**	**44.93**	**43.85**	49.03	48.77	-	15.00
	0	0	10		44.78	36.43	**51.51**	**50.05**	-	14.64
	20	0	0		41.01	23.99	50.27	48.97	4.97	-
	0	20	0		46.87	46.25	50.96	49.38	19.99	14.31
20K	0	0	20	**40K**	44.41	37.13	50.86	49.21	13.68	8.31
	10	10	0		**47.59**	**46.58**	**51.64**	**50.44**	21.81	16.05
	10	0	10		43.47	30.84	51.50	49.51	11.27	6.01
	20	0	0		40.39	22.01	50.43	46.34	-1.50	-
	0	20	0		45.64	**46.69**	49.75	**52.45**	11.31	13.00
40K	0	0	20	**60K**	40.64	31.93	47.86	48.62	-0.90	0.61
	10	10	0		**45.79**	40.10	**51.89**	51.08	11.66	13.37
	10	0	10		42.73	31.30	50.63	48.45	4.21	5.79
	20	0	0		41.71	24.67	51.62	47.97	3.27	-
	0	20	0		45.86	**42.34**	51.19	50.50	13.54	9.95
60K	0	0	20	**80K**	43.27	33.02	50.83	48.39	7.14	3.75
	10	10	0		**46.07**	40.82	**51.93**	**51.17**	14.06	10.45
	10	0	10		42.66	30.58	50.49	47.36	5.63	2.29

Table 3: The table shows the performance of generalist models trained with different data mixtures. **UI** refers to *UltraInteract*, **OO** refers to *OpenOrca*, and **AL** refers to *Alpaca*. The column labeled **Rel. Gain + 20K Data** indicates the relative performance gain of the model in the current row compared to the **UI**-only baseline that uses 20K fewer data points. For example, the performance of a model trained on **40**K data will be compared to the baseline model trained on **20**K **UI** data, as indicated by the blue row above the current data quantity.**Rel. Gain Diver.** denotes the gain of diversification compared to the baseline with the same data budget.

Taori et al., 2023; Peng et al., 2023; Chiang et al., 2023; Xu et al., 2023a; Köksal et al., 2023; Kim et al., 2023). They provide larger instruction sets, at the cost of reduced annotation quality.

Data curation for instruction-tuning. It is widely recognized that the quality of instruction-tuning datasets has a massive impact on the performance of fine-tuned models. Previous works acknowledged the contributions of several key factors. Most research on the subject insist on the importance of the size and quality of the instruction sets (Chung et al., 2022; Iyer et al., 2022; Wang et al., 2023a). Liang et al. (Liang et al., 2024) point out the importance of consistent formats. Several recent works (Zhou et al., 2023a; Cao et al., 2023) suggest that models fine-tuned on carefully selected examples can achieve high performance with small datasets. Various strategies for data curation have been proposed, focusing on instruction diversity, and the quality of answers (Zhou et al., 2023a; Cao et al., 2023; Xu et al., 2023b; Li et al., 2024; Liu et al., 2024b). Several authors discuss the benefit of mixing tasks from different categories (Longpre et al., 2023; Iyer et al., 2022; Bukharin and Zhao, 2024). Closest to our work, Dong et al. (Dong et al., 2024) discuss the impact of mixing general and domain-specific instructions, in order to achieve the best results with the smallest dataset.

7 Conclusion

This work systematically studies instruction-following of language models by modeling it as a computation process and careful controlled experiments. We isolate and study models' instruction-following abilities, offering a new perspective on this fundamental capability. Our experiments further demonstrate that instruction diversity, even within a fixed data budget, plays a critical role in improving model generalization. This finding underscores the value of diverse instruction semantics over large dataset size, in enhancing performance across both specialized and generalized LLM applications. Finally, we offer practical insights into dataset collation strategies, highlighting that proper diversification can significantly outperform dataset expansion.

8 Limitations

This study highlights the impact of dataset diversity on generalization but leaves room for exploring its applicability to broader domains and tasks. While the dynamic data addition process shows promise, refining strategies for balancing domain-specific and general-purpose data could further enhance performance. Future work may also validate these findings across evolving model architectures and real-world settings.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Alexander Bukharin and Tuo Zhao. 2024. Data diversity matters for robust instruction tuning. *Preprint*, arXiv:2311.14736.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction mining: When data mining meets large language model finetuning. *Preprint*, arXiv:2307.06290.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021a. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021b. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. 2023. Theoremqa: A theorem-driven question answering dataset. *Preprint*, arXiv:2305.12524.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. How abilities in large language models are affected by supervised fine-tuning data composition.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. *Preprint*, arXiv:2401.14196.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *Preprint*, arXiv:2212.09689.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, et al. 2022. Opt-iml: Scaling language model instruction meta learning through the lens of generalization. *arXiv preprint arXiv:2212.12017*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Seungone Kim, Se June Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. *Preprint*, arXiv:2305.14045.

Abdullatif Köksal, Timo Schick, Anna Korhonen, and Hinrich Schütze. 2023. Longform: Optimizing instruction tuning for long text generation with corpus extraction. *Preprint*, arXiv:2304.08460.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations – democratizing large language model alignment. *Preprint*, arXiv:2304.07327.

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. 2024. Selective reflection-tuning: Student-selected data recycling for llm instruction-tuning. *Preprint*, arXiv:2402.10110.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023. Openorca: An open dataset of gpt augmented flan reasoning traces. `https://https://huggingface.co/Open-Orca/OpenOrca`.

Shihao Liang, Runchu Tian, Kunlun Zhu, Yujia Qin, Huadong Wang, Xin Cong, Zhiyuan Liu, Xiaojiang Liu, and Maosong Sun. 2024. Exploring format consistency for instruction tuning. *Preprint*, arXiv:2307.15504.

Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, Tianjiao Zhao, Amit Panalkar, Dhagash Mehta, Stefano Pasquali, Wei Cheng, Haoyu Wang, Yanchi Liu, Zhengzhang Chen, Haifeng Chen, Chris White, Quanquan Gu, Jian Pei, Carl Yang, and Liang Zhao. 2024. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *Preprint*, arXiv:2305.18703.

Emmy Liu, Graham Neubig, and Jacob Andreas. 2024a. An incomplete loop: Instruction inference, instruction following, and in-context learning in language models. *Preprint*, arXiv:2404.03028.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Preprint*, arXiv:2305.01210.

Liangxin Liu, Xuebo Liu, Derek F. Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. 2024b. Selectit: Selective instruction tuning for large language models via uncertainty-aware self-reflection. *Preprint*, arXiv:2402.16705.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. *Preprint*, arXiv:2301.13688.

A. A. Markov. 1954. *The theory of algorithms*, volume 42. Acad. Sci. USSR.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *Preprint*, arXiv:2304.03277.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022a. Multitask prompted training enables zero-shot task generalization. *Preprint*, arXiv:2110.08207.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022b. Multitask prompted training enables zero-shot task generalization. *Preprint*, arXiv:2110.08207.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *Preprint*, arXiv:2210.09261.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. Superglue: A stickier benchmark for general-purpose language understanding systems. *Preprint*, arXiv:1905.00537.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023a. How far can camels go? exploring the state of instruction tuning on open resources. *Preprint*, arXiv:2306.04751.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh

Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. *Preprint*, arXiv:2212.10560.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *Preprint*, arXiv:2204.07705.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Source code is all you need. *Preprint*, arXiv:2312.02120.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023a. Wizardlm: Empowering large language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

Yang Xu, Yongqiang Yao, Yufan Huang, Mengnan Qi, Maoquan Wang, Bin Gu, and Neel Sundaresan. 2023b. Rethinking the instruction quality: Lift is what you need. *Preprint*, arXiv:2312.11508.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. *Preprint*, arXiv:2104.08835.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap your own mathematical questions for large language models. *Preprint*, arXiv:2309.12284.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. 2024. Advancing llm reasoning generalists with preference trees. *Preprint*, arXiv:2404.02078.

Liang Zeng, Liangjun Zhong, Liang Zhao, Tianwen Wei, Liu Yang, Jujie He, Cheng Cheng, Rui Hu, Yang Liu, Shuicheng Yan, Han Fang, and Yahui Zhou. 2024. Skywork-math: Data scaling laws for mathematical reasoning in large language models – the story goes on. *Preprint*, arXiv:2407.08348.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024a. When scaling meets LLM finetuning: The effect of data, model and finetuning method. In *The Twelfth International Conference on Learning Representations*.

Xinlu Zhang, Zhiyu Zoey Chen, Xi Ye, Xianjun Yang, Lichang Chen, William Yang Wang, and Linda Ruth Petzold. 2024b. Unveiling the impact of coding data instruction fine-tuning on large language models reasoning. *Preprint*, arXiv:2405.20535.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. Lima: Less is more for alignment. *Preprint*, arXiv:2305.11206.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023b. Instruction-following evaluation for large language models. *Preprint*, arXiv:2311.07911.

A Complement on Markov algorithms

Markov algorithms (Markov, 1954) are ordered sets of rewrite rules, operating on sequences of symbols in a fixed alphabet $\mathcal{U}$. A sequence S is processed by applying the first rewrite applicable to S, at the leftmost position if several exist: i.e. the rewrite rule $ss \rightarrow tr$ transforms the sequence $S = mississipi$ into $S' = mitrissipi$. The algorithm is then applied to S', and the process is repeated until either no rules apply, and the algorithm is said to be *blocked*, or a special rule, called a *stop rule* is invoked, and the algorithm terminates and returns the final rewritten sequence.

Specifically, the algorithm uses and alphabet $\mathcal{A}$, which includes the alphabet $\mathcal{U}$ used buy the sequences to be processed (henceforth, small case latin letters), a set of additional symbols (henceforth, the small case greek letters $\{\alpha, \beta ...\}$, and a special symbol $\cdot$ indicating a stop rule.

For instance, we could define the following algorithm, with $\mathcal{U} = \{a,b\}$, and $\mathcal{A} = \{a,b,\alpha,\beta,\cdot\}$, and the rules

$$\begin{aligned} \alpha x &\rightarrow x\alpha\beta x && (1)\\ \beta x y &\rightarrow y\beta x && (2)\\ \alpha\beta x &\rightarrow x\alpha && (3)\\ \alpha &\rightarrow \cdot && (4)\\ &\rightarrow \alpha && (5) \end{aligned}$$

where x and y stand for any letter a or b. This will transform any sequence of a and b into a concatenation of the sequence and its reverse. Applied on abb, the algorithm will perform the following rewrites:

$$\begin{aligned} abb &\rightarrow \alpha abb && \text{(by 5)}\\ \alpha abb &\rightarrow a\alpha\beta abb && \text{(by 1)}\\ a\alpha\beta abb &\rightarrow a\alpha b\beta ab && \text{(by 2)}\\ a\alpha b\beta ab &\rightarrow ab\alpha\beta b\beta ab && \text{(by 1)}\\ ab\alpha b\beta b\beta ab &\rightarrow ab\alpha\beta bb\beta a && \text{(by 2)}\\ ab\alpha\beta bb\beta a &\rightarrow ab\alpha b\beta b\beta a && \text{(by 2)}\\ ab\alpha b\beta b\beta a &\rightarrow abb\alpha\beta b\beta b\beta a && \text{(by 1)}\\ abb\alpha\beta b\beta b\beta a &\rightarrow abbb\alpha\beta b\beta a && \text{(by 3)}\\ abbb\alpha\beta b\beta a &\rightarrow abbbb\alpha\beta a && \text{(by 3)}\\ abbbb\alpha\beta a &\rightarrow abbbba\alpha && \text{(by 3)}\\ abbbba\alpha &\rightarrow abbbba && \text{(by 4)} \end{aligned}$$

Since rule 4 is a stop rule, the algorithm terminates and returns $abbbba$. Judicious introduction of additional (greek) letters allows one to compose Markov algorithms, effectively writing complex programs. Any effective process (i.e. finite computation) can be represented as a Markov algorithm (this is Markov's thesis).

B Experimental set-up

B.1 Model and Training

In rewrite experiments, we train GPT-2 models (Radford et al., 2019), a decoder-only transformer-based architecture, with 6 layers, 256 dimensions and 4 attention heads from scratch, on a generated instruction-tuning dataset using standard supervised fine-tuning approach. We use the AdamW optimizer, a learning rate of 10^{-3}, and linear scheduling. All models are trained for 50 epochs. For the encrypted-rewriting task, we LoRA fine-tuned Llama-2 models with a learning rate of 1e-5, batch size 64, gradient accumulation step 1, and 8-bit quantization. The model takes about 2000 steps to converge. For coding experiments, we trained the model with a learning rate of 1e-5, batch size 4, and gradient accumulation step 1, 8-bit quantization for 3 epochs with a maximum length of 768. The models are trained and inferenced on 1 Nvidia A40 GPU. We used greedy decoding for all experiments.

B.2 Data Generation

Synthetic Experiment Except for the diversity of semantics experiment, the results we reported in the main paper are obtained from an input length of 50 and a pattern length of 20. To validate the generality of our findings, we conducted experiments on various input sizes {50, 100, 200} and, correspondingly, pattern lengths {20,40,50}.

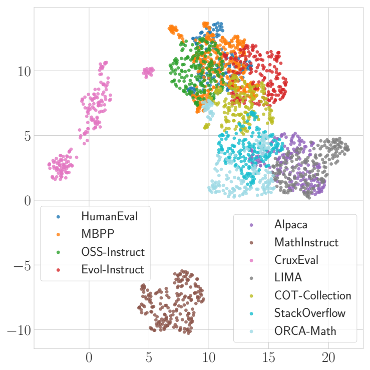
In the diversity of semantics experiment, we used an input length of 500 and a pattern length of 60. We strictly restricted the sub-strings to look for and to replace them with both to be unseen during testing.

Real World Data We downloaded the datasets (OSS-INSTRUCT, ALPACA, COT, ULTRAINTERACT, OPENORCA) from the official Huggingface Datasets Repos.

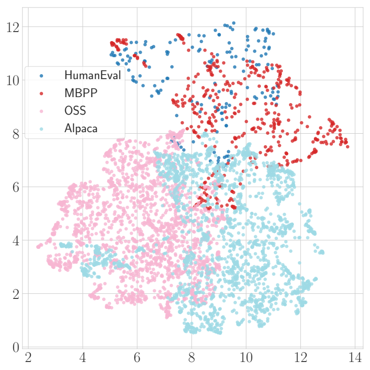
Demonstration of dataset sizes for long-tail rule distribution experiments. We included the plot of percentage against rank index with different α's.

C More Details On Evaluation

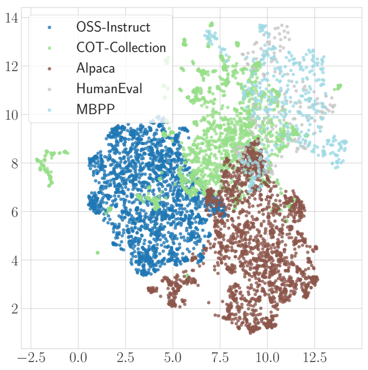
For coding task, we evaluated the performance following the standard settings in EvalPlus (Liu et al., 2023).



(a) Semantic clustering of relevant datasets.



(b) OSS-Alpaca mixture and test instructions.



(c) OSS-COT-Alpaca mixture and test instructions.

Figure 7: Visualization of embedded instructions.

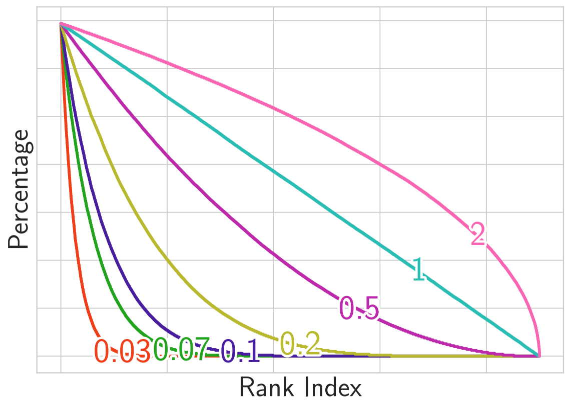


Figure 8: The sorted percentage of each instruction following power-law distribution with different shape parameters. The y-axis is the percentage of the rules in the training mixture. The x-axis is the ranked index (by proportion of examples) of instructions.

In Section 5, we evaluated a on variety of tasks. We used the evaluation suite (prompt, score computation script) provided by Yuan et al. (Yuan et al., 2024).